

## Chapter 5

## Conclusions



# Bibliography

- [1] Romano,A., Lehmann,D., Pradel,M., and Wang,W. (2022). Wobfuscator: Obfuscating javascript malware via opportunistic translation to webassembly. In *2022 IEEE Symposium on Security and Privacy (SP) (SP)*, pages 1101–1116, Los Alamitos, CA, USA. IEEE Computer Society.
- [2] Voulimeneas,A., Song,D., Larsen,P., Franz,M., and Volckaert,S. (2021). dmvx: Secure and efficient multi-variant execution in a distributed setting. In *Proceedings of the 14th European Workshop on Systems Security*, pages 41–47.
- [3] Narayan,S., Disselkoe,C., Moghimi,D., Cauligi,S., Johnson,E., Gang,Z., Vahldiek-Oberwagner,A., Sahita,R., Shacham,H., Tullsen,D., et al. (2021). Swivel: Hardening webassembly against spectre. In *USENIX Security Symposium*.
- [4] Lee,S., Kang,H., Jang,J., and Kang,B. B. (2021). Savior: Thwarting stack-based memory safety violations by randomizing stack layout. *IEEE Transactions on Dependable and Secure Computing*.
- [5] Johnson,E., Thien,D., Alhessi,Y., Narayan,S., Brown,F., Lerner,S., McMullen,T., Savage,S., and Stefan,D. (2021). Sfi safety for native-compiled wasm. *NDSS. Internet Society*.
- [6] Hilbig,A., Lehmann,D., and Pradel,M. (2021). An empirical study of real-world webassembly binaries: Security, languages, use cases. *Proceedings of the Web Conference 2021*.
- [7] Cabrera Arteaga,J., Laperdrix,P., Monperrus,M., and Baudry,B. (2021b). Multi-Variant Execution at the Edge. *arXiv e-prints*, page arXiv:2108.08125.
- [8] Cabrera Arteaga,J., Laperdrix,P., Monperrus,M., and Baudry,B. (2021a). Multi-Variant Execution at the Edge. *arXiv e-prints*, page arXiv:2108.08125.
- [9] Cabrera Arteaga,J., Floros,O., Vera Perez,O., Baudry,B., and Monperrus,M. (2021). Crow: code diversification for webassembly. In *MADWeb, NDSS 2021*.
- [10] (2021). National Cyber Leap Year.

- [11] Xu,Y., Solihin,Y., and Shen,X. (2020). Merr: Improving security of persistent memory objects via efficient memory exposure reduction and randomization. In *Proc. of ASPLOS*, pages 987–1000.
- [12] Shillaker,S. and Pietzuch,P. (2020). Faasm: Lightweight isolation for efficient stateful serverless computing. In *USENIX Annual Technical Conference*, pages 419–433.
- [13] Runeson,P., Engström,E., and Storey,M.-A. (2020). *The Design Science Paradigm as a Frame for Empirical Software Engineering*, pages 127–147. Springer International Publishing, Cham.
- [14] Lehmann,D., Kinder,J., and Pradel,M. (2020). Everything old is new again: Binary security of webassembly. In *29th USENIX Security Symposium (USENIX Security 20)*. USENIX Association.
- [15] Harrand,N., Soto-Valero,C., Monperrus,M., and Baudry,B. (2020). Java decompiler diversity and its application to meta-decompilation. *Journal of Systems and Software*, 168:110645.
- [16] Gadepalli,P. K., McBride,S., Peach,G., Cherkasova,L., and Parmer,G. (2020). Sledge: A serverless-first, light-weight wasm runtime for the edge. In *Proceedings of the 21st International Middleware Conference*, page 265–279.
- [17] Chen,D. and W3C group (2020). WebAssembly documentation: Security. Accessed: 18 June 2020.
- [18] Bryant,D. (2020). Webassembly outside the browser: A new foundation for pervasive computing. In *Proc. of ICWE 2020*, pages 9–12.
- [19] Roy,A., Chhabra,A., Kamhoua,C. A., and Mohapatra,P. (2019). A moving target defense against adversarial machine learning. In *Proceedings of the 4th ACM/IEEE Symposium on Edge Computing*, page 383–388.
- [20] Österlund,S., Koning,K., Olivier,P., Barbalace,A., Bos,H., and Giuffrida,C. (2019). kmvx: Detecting kernel information leaks with multi-variant execution. In *ASPLOS*.
- [21] Aga,M. T. and Austin,T. (2019). Smokestack: thwarting dop attacks with runtime stack layout randomization. In *Proc. of CGO*, pages 26–36.
- [22] Silvanovich,N. (2018). The problems and promise of webassembly. Technical report.
- [23] Lu,K., Xu,M., Song,C., Kim,T., and Lee,W. (2018). Stopping memory disclosures via diversification and replicated execution. *IEEE Transactions on Dependable and Secure Computing*.

- [24] Li,J., Zhao,B., and Zhang,C. (2018). Fuzzing: a survey. *Cybersecurity*, 1(1):1–13.
- [25] Genkin,D., Pachmanov,L., Tromer,E., and Yarom,Y. (2018). Drive-by key-extraction cache attacks from portable code. *IACR Cryptol. ePrint Arch.*, 2018:119.
- [26] Belleville,N., Couroussé,D., Heydemann,K., and Charles,H.-P. (2018). Automated software protection for the masses against side-channel attacks. *ACM Trans. Archit. Code Optim.*, 15(4).
- [27] WebAssembly Community Group (2017b). WebAssembly Specification.
- [28] WebAssembly Community Group (2017a). WebAssembly Roadmap.
- [29] Sengupta,S., Vadlamudi,S. G., Kambhampati,S., Doupe,A., Zhao,Z., Taguinod,M., and Ahn,G.-J. (2017). A game theoretic approach to strategy generation for moving target defense in web applications. In *Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems*, page 178–186.
- [30] Haas,A., Rossberg,A., Schuff,D. L., Schuff,D. L., Titzer,B. L., Holman,M., Gohman,D., Wagner,L., Zakai,A., and Bastien,J. F. (2017). Bringing the web up to speed with webassembly. *PLDI*.
- [31] Van Es,N., Nicolay,J., Stievenart,Q., D’Hondt,T., and De Roover,C. (2016). A performant scheme interpreter in asm.js. In *Proceedings of the 31st Annual ACM Symposium on Applied Computing, SAC ’16*, page 1944–1951, New York, NY, USA. Association for Computing Machinery.
- [32] Koning,K., Bos,H., and Giuffrida,C. (2016). Secure and efficient multi-variant execution using hardware-assisted process virtualization. In *2016 46th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, pages 431–442. IEEE.
- [33] Couroussé,D., Barry,T., Robisson,B., Jaillon,P., Potin,O., and Lanet,J.-L. (2016). Runtime code polymorphism as a protection against side channel attacks. In *IFIP International Conference on Information Security Theory and Practice*, pages 136–152. Springer.
- [34] Volckaert,S., Coppens,B., and De Sutter,B. (2015). Cloning your gadgets: Complete rop attack immunity with multi-variant execution. *IEEE Transactions on Dependable and Secure Computing*, 13(4).
- [35] Morgan,T. D. and Morgan,J. W. (2015). Web timing attacks made practical. *Black Hat*.
- [36] Kim,D., Kwon,Y., Sumner,W. N., Zhang,X., and Xu,D. (2015). Dual execution for on the fly fine grained execution comparison. *SIGPLAN Not.*

- [37] Davi, L., Liebchen, C., Sadeghi, A.-R., Snow, K. Z., and Monrose, F. (2015). Isomeron: Code randomization resilient to (just-in-time) return-oriented programming. In *NDSS*.
- [38] Crane, S., Homescu, A., Brunthaler, S., Larsen, P., and Franz, M. (2015). Thwarting cache side-channel attacks through dynamic software diversity. In *NDSS*, pages 8–11.
- [39] Baudry, B. and Monperrus, M. (2015). The multiple facets of software diversity: Recent developments in year 2000 and beyond. *ACM Comput. Surv.*, 48(1).
- [40] Alon Zakai (2015). asm.js Speedups Everywhere.
- [41] Agosta, G., Barengi, A., Pelosi, G., and Scandale, M. (2015). The MEET approach: Securing cryptographic embedded software against side channel attacks. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 34(8):1320–1333.
- [42] Okhravi, H., Rabe, M., Mayberry, T., Leonard, W., Hobson, T., Bigelow, D., and Streilein, W. (2013). Survey of cyber moving targets. *Massachusetts Inst of Technology Lexington Lincoln Lab, No. MIT/LL-TR-1166*.
- [43] Homescu, A., Neisius, S., Larsen, P., Brunthaler, S., and Franz, M. (2013). Profile-guided automated software diversity. In *Proceedings of the 2013 IEEE/ACM International Symposium on Code Generation and Optimization (CGO)*, pages 1–11. IEEE.
- [44] Coppens, B., De Sutter, B., and Maebe, J. (2013). Feedback-driven binary code diversification. *ACM Transactions on Architecture and Code Optimization (TACO)*, 9(4):1–26.
- [45] Maurer, M. and Brumley, D. (2012). Tachyon: Tandem execution for efficient live patch testing. In *21st USENIX Security Symposium (USENIX Security 12)*, pages 617–630.
- [46] Jackson, T. (2012). *On the Design, Implications, and Effects of Implementing Software Diversity for Security*. PhD thesis, University of California, Irvine.
- [47] Sidiroglou-Douskos, S., Misailovic, S., Hoffmann, H., and Rinard, M. (2011). Managing performance vs. accuracy trade-offs with loop perforation. In *Proceedings of the 19th ACM SIGSOFT Symposium and the 13th European Conference on Foundations of Software Engineering, ESEC/FSE '11*, page 124–134, New York, NY, USA. Association for Computing Machinery.
- [48] Salamat, B., Jackson, T., Wagner, G., Wimmer, C., and Franz, M. (2011). Runtime defense against code injection attacks using replicated execution. *IEEE Trans. Dependable Secur. Comput.*, 8(4):588–601.

- [49] Amarilli,A., Müller,S., Naccache,D., Page,D., Rauzy,P., and Tunstall,M. (2011). Can code polymorphism limit information leakage? In *IFIP International Workshop on Information Security Theory and Practices*, pages 1–21. Springer.
- [50] Chen,T. Y., Kuo,F.-C., Merkel,R. G., and Tse,T. H. (2010). Adaptive random testing: The art of test case diversity. *J. Syst. Softw.*, 83:60–66.
- [51] Salamat,B., Jackson,T., Gal,A., and Franz,M. (2009). Orchestra: intrusion detection using parallel execution and monitoring of program variants in user-space. In *Proceedings of the 4th ACM European conference on Computer systems*, pages 33–46.
- [52] Maia,M. D. A., Sobreira,V., Paixão,K. R., Amo,R. A. D., and Silva,I. R. (2008). Using a sequence alignment algorithm to identify specific and common code from execution traces. In *Proceedings of the 4th International Workshop on Program Comprehension through Dynamic Analysis (PCODA)*, pages 6–10.
- [53] Jacob,M., Jakubowski,M. H., Naldurg,P., Saw,C. W. N., and Venkatesan,R. (2008). The superdiversifier: Peephole individualization for software protection. In *International Workshop on Security*, pages 100–120. Springer.
- [54] Salamat,B., Gal,A., Jackson,T., Manivannan,K., Wagner,G., and Franz,M. (2007). Stopping buffer overflow attacks at run-time: Simultaneous multi-variant program execution on a multicore processor. Technical report, Technical Report 07-13, School of Information and Computer Sciences, UCIrvine.
- [55] Bruschi,D., Cavallaro,L., and Lanzi,A. (2007). Diversified process replicæ for defeating memory error exploits. In *Proc. of the Int. Performance, Computing, and Communications Conference*.
- [56] Cox,B., Evans,D., Filipi,A., Rowanhill,J., Hu,W., Davidson,J., Knight,J., Nguyen-Tuong,A., and Hiser,J. (2006). N-variant systems: a secretless framework for security through diversity. In *Proc. of USENIX Security Symposium*, USENIX-SS’06.
- [57] Pohl,K., Böckle,G., and Van Der Linden,F. (2005). *Software product line engineering: foundations, principles, and techniques*, volume 1. Springer.
- [58] Bhatkar,S., Sekar,R., and DuVarney,D. C. (2005). Efficient techniques for comprehensive protection from memory error exploits. In *Proceedings of the USENIX Security Symposium*, pages 271–286.
- [59] Kc,G. S., Keromytis,A. D., and Prevelakis,V. (2003). Countering code-injection attacks with instruction-set randomization. In *Proc. of CCS*, pages 272–280.

- [60] Bhatkar,S., DuVarney,D. C., and Sekar,R. (2003). Address obfuscation: an efficient approach to combat a board range of memory error exploits. In *Proceedings of the USENIX Security Symposium*.
- [61] Barrantes,E. G., Ackley,D. H., Forrest,S., Palmer,T. S., Stefanovic,D., and Zovi,D. D. (2003). Randomized instruction set emulation to disrupt binary code injection attacks. In *Proc. CCS*, pages 281–289.
- [62] Chew,M. and Song,D. (2002). Mitigating buffer overflows by operating system randomization. Technical Report CS-02-197, Carnegie Mellon University.
- [63] Forrest,S., Somayaji,A., and Ackley,D. (1997). Building diverse computer systems. In *Proceedings. The Sixth Workshop on Hot Topics in Operating Systems (Cat. No.97TB100133)*, pages 67–72.
- [64] Cohen,F. B. (1993). Operating system protection through program evolution. *Computers & Security*, 12(6):565–584.
- [65] Avizienis and Kelly (1984). Fault tolerance by design diversity: Concepts and experiments. *Computer*, 17(8):67–80.
- [66] Needleman,S. B. and Wunsch,C. D. (1970). A general method applicable to the search for similarities in the amino acid sequence of two proteins. 48(3):443–453.
- [67] Gnanadesikan,R. and Wilk,M. B. (1968). Probability plotting methods for the analysis of data. *Biometrika*, 55(1):1–17.
- [68] Mann,H. B. and Whitney,D. R. (1947). On a test of whether one of two random variables is stochastically larger than the other. *Ann. Math. Statist.*, 18(1):50–60.